ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Trajectory-Based Spatiotemporal Entity Linking

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Jin F., Hua W., Zhou T., Xu J., Francia M., Orowska M., et al. (2022). Trajectory-Based Spatiotemporal Entity Linking. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 34(9), 4499-4513 [10.1109/TKDE.2020.3036633].

Availability:
This version is available at: https://hdl.handle.net/11585/789283 since: 2023-01-11

Published:
DOI: http://doi.org/10.1109/TKDE.2020.3036633

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

12 October 2024

This is the final peer-reviewed accepted manuscript of:

F. Jin et al., "Trajectory-Based Spatiotemporal Entity Linking," in IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 9, pp. 4499-4513, 1 Sept. 2022

The final published version is available online at ***https://dx.doi.org/10.1109/TKDE.2020.3036633***

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

Trajectory-Based Spatiotemporal Entity Linking

Fengmei Jin, Wen Hua, Thomas Zhou, Jiajie Xu, Matteo Francia, Maria E Orlowska, Xiaofang Zhou, *Fellow, IEEE*

Abstract—Trajectory-based spatiotemporal entity linking is to match the same moving object in different datasets based on their movement traces. It is a fundamental step to support spatiotemporal data integration and analysis. In this paper, we study the problem of spatiotemporal entity linking using effective and concise signatures extracted from their trajectories. This linking problem is formalized as a k-nearest neighbor (k-NN) query on the signatures. Four representation strategies (sequential, temporal, spatial, and spatiotemporal) and two quantitative criteria (commonality and unicity) are investigated for signature construction. A simple yet effective dimension reduction strategy is developed together with a novel indexing structure called the WR-tree to speed up the search. A number of optimization methods are proposed to improve the accuracy and robustness of the linking. Our extensive experiments on real-world datasets verify the superiority of our approach over the state-of-the-art solutions in terms of both accuracy and efficiency.

Index Terms—spatiotemporal entity linking, moving objects, signature, dimension reduction, k-NN search, weighted R-tree

1 Introduction

With the prevalence of location-capturing devices and location-based services comes an ever-increasing amount and variety of spatial trajectory data, such as vehicle trajectories, sensor readings, telecom tokens, and location check-ins. The mining of mobility patterns has become an important research topic due to its utility in significant real-world applications including transport management and urban planning. One interesting problem to study is the extent to which individual movements are unique and distinctive, i.e., the possibility of identifying an individual based on her movement patterns extracted from the historical traces. Such study can benefit various real-world applications.

Scenario 1: A user may have multiple social network accounts. By matching accounts across platforms (e.g., Twitter and Facebook), a more comprehensive user profile can be constructed, which potentially improves the performance of personalized recommendation.

Scenario 2: User monitoring is possible by linking phone numbers used by a single person. This is extremely important for security control such as criminal tracking.

Scenario 3: A taxi driver can register with multiple Uber-like companies. Once this information is combined together through entity linking, different driving behaviors of the same user could be observed, stimulating transformation of operation patterns in the company.

A recent study on mobile phone users [1] has shown that the uniqueness of human mobility is high enough that 4 randomly-sampled spatiotemporal points could uniquely identify 95% of individuals. However, can the same method be utilized to identify other types of moving objects? We conducted a preliminary experimental study on a real-world taxi dataset, and observed that only 15% of taxis could be identified using the method proposed in [1]. This motivated us to explore a better representation of mobility patterns for accurate moving object linking. Unlike traditional methods which discover mobility patterns to capture human's collective [2] [3], sequential [4] [5] or periodic [6] [7] movement behaviors, in this work, we aim at extracting patterns, hereafter referred to as *signatures*, for entity linking, i.e., matching traces made by the same entity in different datasets such as social media users and taxi drivers.

1.1 Challenges and Contributions

How to effectively represent and quantify the signature of a moving object to guarantee the accuracy of object linking? Trajectories are intrinsically spatial, temporal, and sequential. Hence, the object signature should also capture these characteristics. Existing trajectory pattern mining algorithms [4] [5] [6] [7] extract sequential and temporal patterns from trajectories, but ignore spatial features. However, our empirical study on a real-world taxi dataset shows that spatial patterns are the most effective for identifying individuals, as opposed to the other two features. Additionally, signatures need to encode patterns quantitatively to allow for similarity calculations, which is a non-trivial task. Intuitively, the quantitative signature should be able to encapsulate behavior that is not only ubiquitous throughout the whole trace of the individual but also highly discriminative from one individual to another. In this paper, two metrics are proposed to quantify a signature from an individual's historical trace: 1) *commonality*, the frequency of the signature in the individual's trace, and 2) *unicity*, the extent to which an individual can be distinguished from others by their signatures.

How to improve the efficiency of moving object linking? This work reduces the task of object linking to a k-nearest neighbor (k-NN) search problem: given a query object, we search in a set of moving objects to find the top-k candidates based on signature similarity. The most straightforward approach is a linear scan which calculates all pairwise similarities one by one. However, this is infeasible in practice due to the following challenges:

Curse of dimensionality: The signatures can be extremely large. In the worst case, when every point in an individual's historical trace contributes to her personalized profile, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

size of signatures could be as large as the number of distinct points in the entire dataset (or even larger if sequential patterns are considered). The computational complexity of comparing two signatures is $O(d)$ where d represents the dimensionality of the signatures; When d is large, successive comparisons will collectively incur an enormous time cost. Although dimension reduction has been extensively studied in the literature (e.g., PCA [8], LSH [9] [10], etc.), later experiments show that existing methods significantly degrade the linking accuracy when the dimensionality is highly reduced.

Curse of cardinality: In practice, the cardinality of the object set, denoted as n, is usually massive: reaching millions of candidates. The complexity of the aforementioned pairwise checking method is $O(n \times d)$ which includes lots of unnecessary calculations, since most candidates are actually unpromising in regards to being a member of the k-nearest neighbors for the query object. Various algorithms (e.g., *AllPairs* [11], *APT* [12], *MMJoin* [13], *L2AP* [14]) and indexing structures (e.g., *k-d tree* [15] [16] [17], *R-tree* [18] [19] [20], *hB-tree* [21] [22]) have been proposed to scale up the similarity search or k-NN search. Nevertheless, later experiments show that directly applying these methods can only achieve limited efficiency improvements.

Our major contributions in this paper include:

- An effective way to generate signatures from an object's moving history is proposed. Four representation strategies (i.e., sequential, temporal, spatial, and spatiotemporal) and two quantitative metrics (i.e., commonality and unicity) are studied for their effectiveness in signature construction. High accuracy is achieved for taxi trajectories, which are commonly believed as hard to personalize.
- A simple yet effective signature reduction method, CUT, is developed with significantly better performance than the existing algorithms including PCA and LSH based on *spatial shrinking* of moving object footprints.
- The high-dimensional object linking problem is transformed to a k-NN search in 2D space. A novel indexing structure, WR-tree, is designed to speed up k-NN search on signatures. A bulk-loading index construction method is introduced based on a novel optimization criterion of signature enlargement. This index mechanism can support updates (i.e., adding new objects).
- Two optimization schemes, re-ranking and stable marriage, are proposed to refine the k-NN results and improve the robustness of the linking algorithm.
- Extensive experiments are conducted on a real-world taxi dataset, demonstrating significantly better accuracy and efficiency by our proposal compared with the state-of-the-art approaches. Object linkability is also shown to be highly sensitive to the signatures, which opens a new approach to trajectory privacy protection.

2 Problem Statement

This section introduces some preliminary concepts and formally defines the problem of spatiotemporal entity linking. Table 1 summarizes several key notations.

Definition 2.1 (Spatiotemporal Entity)**.** A spatiotemporal entity, denoted as o, is a moving object characterized by its position in space that varies over time.

TABLE 1
Summarization of key notations.

Notation	Definition
o	a moving object
$T(o)$	the historical trace of a moving object o
$f(o)$	the (reduced) signature of a moving object o. $d = \|f(o)\|$ represents the dimensionality of the signature
$sim(o_1, o_2)$	the signature similarity between objects o_1 and o_2
kNN(q, O)	the k-nearest neighbors of query object q in the candidate object set O, and $n = \|O\|$ represents the cardinality of the candidate object set
$\langle g, n(g)\rangle$	g is a q-gram and $n(g)$ is the frequency of g in object o's sequential signature $f(o)$
$\langle T, n(T)\rangle$	T is a time interval and $n(T)$ is the frequency of T in object o's temporal signature $f(o)$
$\langle p, w(p)\rangle$	p is a spatial point and $w(p)$ is the TF-IDF weight of p in object o's spatial signature $f(o)$
$\langle h, w(h)\rangle$	$h = (p, T)$ consists of a spatial location p and a time interval T, and $w(h)$ is the TF-IDF weight of h in object o's spatiotemporal signature $f(o)$
$MBR(o)$	the minimum bounding rectangle over object o's spatial signature $f(o)$

Definition 2.2 (Trace)**.** The trace of a spatiotemporal entity (or "moving object", interchangeably) represents its entire movement history. It is a sequence of spatiotemporal points, denoted as $T(o) = \langle p_1, p_2, \ldots, p_n\rangle$ where each $p = (x, y, t)$ consists of a location (x, y) (e.g., longitude and latitude) at time t. Points in a trace are organized chronologically, namely $\forall i < j : p_i.t < p_j.t$.

The collection of all moving objects is denoted as $O = \{o_1, o_2, \ldots, o_n\}$ with its cardinality $n = |O|$. The historical trace of a moving object o can to some extent reflect o's personalized movement pattern, which is valuable for object identification. Logically, two objects with different IDs are possibly the same real-world entity if they share highly similar movement patterns (i.e., *signatures*).

Definition 2.3 (Signature)**.** The signature of a moving object o, denoted as $f(o)$, is a quantitative representation of o's movement pattern, such that the similarity between two objects o_1 and o_2 can be measured by the similarity between their corresponding signatures $f(o_1)$ and $f(o_2)$.

It is assumed that each point in object o's trace $T(o)$ contributes partially to the object's personalized profile: $f(o)$ is constructed from o's entire movement history. Various strategies are proposed to quantify signatures $f(o)$ and calculate signature similarity between two objects $sim(o_1, o_2)$ (Section 3). For representation simplicity, $f(o)$ is used hereafter to denote all types of signatures.

Given two sets of moving objects along with their signatures, we aim to identify all pairs of objects that possibly refer to the same real-world entity based on signature similarities. We reduce this task to a k-nearest neighbor (k-NN) search problem. In other words, for each object in one dataset, a k-NN search is conducted in the other dataset to find the matched object.

Definition 2.4 (Spatiotemporal Entity Linking)**.** Given a query object q and a collection of moving objects $O = \{o_1, o_2, \ldots, o_n\}$, find the top-$k$ nearest neighbors for q with the largest signature similarity $sim(q, o)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

3 SIGNATURE

Inspired by the unique characteristics of trajectories, we introduce various methods for signature representation from an object's moving history and propose two heuristic metrics to quantify each signature, as described below:

Commonality. The signature should be representative of an object's movement profile, i.e., ubiquitous throughout the whole trace of the moving object. We define commonality as the frequency of the signature in the object's historical trace.

Unicity. The signature represents an object's personalized movement profile, and should be highly discriminative between candidate objects. We define unicity as the number of unique objects containing the signature, which quantifies the signature's ability to distinguish an object from others.

Given the quantitative signatures of objects, we can estimate their similarities accordingly. We will discuss the technical details in the following sections.

3.1 Signature Representation

1) Sequential Signature

Moving objects visit locations in a particular order such that their sequential behavior can become an identifying feature of their historical trajectory. Many existing works on trajectory pattern mining focus on identifying sequential patterns (i.e., a common sequence of locations) from a set of trajectories [4] [5], which are then used for real-world applications including routing, location prediction, traffic analysis, etc. This work explores the possibility of utilizing sequential behavior for moving object linking. Specifically, an object o's historical trace $T(o)$ can be regarded as a document, and a set of q-grams is extracted from $T(o)$. Each q-gram can be associated with its weight in $T(o)$ to identify the most representative and distinctive sequential behaviors of the moving object.

Definition 3.1 (Sequential Signature)**.** The sequential signature of an object o is a collection of weighted q-grams extracted from its historical trace $T(o)$. Specifically, $f(o) = \{\langle g_1, w(g_1)\rangle, \langle g_2, w(g_2)\rangle, \ldots, \langle g_d, w(g_d)\rangle\}$, where g is a q-gram (i.e., a subsequence of q consecutive points), $w(g)$ is its weight in $T(o)$, and d is the total number of distinct q-grams in the entire dataset (i.e., *the dimensionality* of $f(o)$) which could be extremely large in practice. We adopt the TF-IDF weighing strategy to calculate $w(g)$, which can capture both commonality and unicity of the sequential signature.

Given two moving objects o_1 and o_2 along with their sequential signatures $f(o_1)$ and $f(o_2)$, their similarity is measured using the cosine similarity between the corresponding signatures. To simplify computation, we transform sequential signature into unit length by L_2 normalization and then calculate signature similarity by the dot product, i.e.,

$$\begin{aligned} sim(o_1, o_2) = \cos(f(o_1), f(o_2)) &= \frac{f(o_1) \cdot f(o_2)}{||f(o_1)|| \times ||f(o_2)||} \\ &= f(o_1) \cdot f(o_2) = \sum_{i=1}^{d} w_{o_1}(g_i) \times w_{o_2}(g_i) \end{aligned} \tag{1}$$

where $w_o(g)$ represents the normalized weight of q-gram g in signature $f(o)$. In the rest of the paper, we use $f(o)$ to denote the L_2 normalized sequential signature of object o.

2) Temporal Signature

An individual's moving behavior exhibits some temporal patterns. Consider vehicles: Private cars usually travel at commuter time daily (e.g., 9am and 5pm) whereas taxis might run uninterruptedly all day; Some taxi drivers carry passengers during the daytime whereas others prefer to operate at night. Evidently, temporal travel patterns can help to distinguish moving objects to some extent, motivating the following definition of a temporal signature.

Definition 3.2 (Temporal Signature)**.** The temporal signature of a moving object o is a histogram over equal-size time intervals: $f(o) = [\langle T_1, n(T_1)\rangle, \langle T_2, n(T_2)\rangle, \ldots, \langle T_d, n(T_d)\rangle]$. In particular, we divide one day into d intervals (or bins) where d is the dimensionality of the temporal signature. Each T_i represents a time interval of length $\Delta t = \frac{24}{d}$ hours, and $n(T_i)$ counts the total number of points in o's historical trace $T(o)$ whose timestamp falls into this interval, i.e., $n(T_i) = |\{p | p \in T(o) \wedge p.t \in T_i\}|$.

The similarity between two histograms can be measured by various methods such as correlation, intersection, Chi-square, Battachary distance, KL divergence, L_p-norm distance, Earth mover's distance (EMD), etc. We adopt EMD [23] in this work as it further takes into consideration cross-bin information rather than simply conducting a bin-to-bin matching. In order to apply EMD, we transform the temporal signature via L_1 normalization (i.e., $n(T_i) = \frac{n(T_i)}{\sum_{j=1}^{d} n(T_j)}$) as EMD requires the two histograms have the same integral. EMD measures the minimum cost ($c_{i,j}$) to transform one histogram to the other by transporting elements between bins. Intuitively, the larger the distance between i-th bin and j-th bin, the larger the cost $c_{i,j}$. However, the distance between temporal intervals cannot be measured by the number of bins in between. For example, 1:00-2:00 is temporally close to 23:00-24:00 while their bin-wise distance is quite large. Therefore, we design the cost $c_{i,j}$ as below:

$$c_{i,j} = \begin{cases} \frac{|i-j| \times \Delta t}{12}, & \textit{if } |i-j| \times \Delta t \leq 12 \\ \frac{24 - |i-j| \times \Delta t}{12}, & \textit{otherwise} \end{cases}$$

Given two objects o_1 and o_2 with corresponding temporal signatures $f(o_1)$ and $f(o_2)$, we measure their similarity as:

$$sim(o_1, o_2) = 1 - emd(f(o_1), f(o_2)) \tag{2}$$

where $emd(f(o_1), f(o_2))$ is the EMD distance between the two histograms $f(o_1)$ and $f(o_2)$.

3) Spatial Signature

In general, a trajectory can be regarded as a special type of time series data whose element represents a geographical location rather than a numeric value. Therefore, besides the sequential and temporal features that exist in general time series, spatial information is also relevant for moving object identification. That is, different moving objects could have different preferences for the locations they visit (residence, restaurant, scenic spot, gas station, etc.). As discovered in [1], 4 locations can identify 95% of the mobile phone users.

Definition 3.3 (Spatial Signature)**.** We define the spatial signature of a moving object o as a weighted vector over the points in its entire historical trace $T(o)$. More specifically,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

$f(o) = (\langle p_1, w(p_1)\rangle, \langle p_2, w(p_2)\rangle, \ldots, \langle p_d, w(p_d)\rangle)$, where d represents the total number of distinct points in the entire dataset (i.e., the dimensionality of $f(o)$) and $w(p)$ is the weight of point p reflecting its representativeness and distinctiveness.

In this work, we adopt the TF-IDF weighing strategy to construct the spatial signature $f(o)$ of moving object o and estimate the importance of each point p in $f(o)$:

$$w(p) = tf(p) \times idf(p)$$

- **Commonality** measures the frequency of a point in the object's trace, i.e., $tf(p) = \frac{N_{p,T(o)}}{|T(o)|}$ where $N_{p,T(o)}$ is the total number of times p occurs in $T(o)$ and $|T(o)|$ is the total number of points contained in $T(o)$.
- **Unicity** measures how much discriminative information a point provides, i.e., $idf(p) = \log \frac{|O|}{|O_p|}$ where $|O| = n$ represents the total number of moving objects (i.e., the cardinality of candidate object set O) and $|O_p|$ represents the number of objects containing p in their traces, namely $O_p = \{o | p \in T(o)\}$.

Given two moving objects o_1 and o_2, their similarity can be measured by the cosine similarity between the corresponding spatial signatures $f(o_1)$ and $f(o_2)$. To simplify the computation, we transform each spatial signature into unit length by L_2 normalization and then calculate the signature similarity by the dot product, i.e.,

$$\begin{aligned} sim(o_1, o_2) = \cos(f(o_1), f(o_2)) &= \frac{f(o_1) \cdot f(o_2)}{||f(o_1)|| \times ||f(o_2)||} \\ &= f(o_1) \cdot f(o_2) = \sum_{i=1}^{d} w_{o_1}(p_i) \times w_{o_2}(p_i) \end{aligned} \tag{3}$$

where $w_o(p)$ represents the normalized weight of point p in signature $f(o)$. In the rest of the paper, we still use $f(o)$ to denote the L_2 normalized spatial signature of object o.

4) Spatiotemporal Signature

Naturally, we can combine multiple features to see if it can achieve better performance. Our empirical results show that sequential signatures are less valuable for object linking. Therefore, we only consider combining spatial and temporal features in this work, i.e., the spatiotemporal signature.

Definition 3.4 (Spatiotemporal Signature)**.** We define the spatiotemporal signature of a moving object o as a weighted vector over spatiotemporal dimensions $h = (p, T)$, where p represents a spatial location and T is a time interval described in Definition 3.2. We denote the signature as $f(o) = (\langle h_1, w(h_1)\rangle, \langle h_2, w(h_2)\rangle, \ldots, \langle h_d, w(h_d)\rangle)$.

As in Definition 3.3, we adopt the TF-IDF weighting strategy to calculate $w(h)$. The only difference is that we need to count the total number of times location p occurs in object o's trace $T(o)$ during the time interval of T. Combining the two signatures intuitively enlarges the feature space: to alleviate the sparsity of spatiotemporal dimensions, we construct the signature at a coarser granularity via a grid-based spatial representation. That is, p in $h = (p, T)$ represents a grid in the spatiotemporal signature.

All four features (sequential, temporal, spatial, and spatiotemporal) are commonly believed to be very important for various applications of moving object modeling. In this work, we empirically evaluate the four signatures on a real-world taxi dataset. Interestingly, sequential, temporal, and spatiotemporal signatures are not as effective as the spatial signature for moving object linking. Therefore, we will only consider $f(o)$ as the spatial signature hereafter.

3.2 Signature Reduction

Recall that $f(o) = (\langle p_1, w(p_1)\rangle, \langle p_2, w(p_2)\rangle, \ldots, \langle p_d, w(p_d)\rangle)$. The dimensionality d of the original signature is practically huge considering the large number of points contained in trajectories. This makes it very time-consuming to calculate pairwise signature similarity. Various techniques have been proposed for dimensionality reduction, including principal component analysis (PCA), locality-sensitive hash (LSH), which can be applied to speed up the calculation of cosine similarity and obtain approximately similar signatures.

Both PCA [8] and LSH [24] [25] are applied in this work for signature reduction in order to speed up the cosine similarity calculation. However, our experimental results in Section 6.2.2 reveal that we cannot achieve a satisfactory object linking using PCA or LSH if we reduce the signature dimensionality too much (e.g., 10 to 100 dimensions). Fortunately, we observe that signatures exhibit a power-law distribution, meaning that only a small number of points contribute to the majority of the total weights. Therefore, we propose the CUT method (i.e., cutting the long tail) which reduces the original signature by reserving its top-m weighted points only (i.e., updating the weights of all the other points to 0). According to the empirical evaluation in Section 6.2.2, such a simple approach to signature reduction can achieve an average accuracy of 80.6% for taxi linking when the signature dimensionality is reduced from 160,000 to 10 (5% performance degradation), outperforming PCA and LSH by 79.9% and 76% respectively.

The CUT method for dimension reduction naturally results in the side effect of *spatial shrinking* for signatures: the original signature could cover a huge spatial region (vehicles, in particular taxis, could potentially traverse the whole city), while the reduced signature after cutting the long tail is scattered in several small regions (workplace, shopping malls, residence, etc.), since the CUT method only preserves unique locations visited frequently by a moving object. To verify this, we conduct a statistical analysis on a real-world taxi dataset, which compares the pairwise spatial overlap ratio between original signatures with the ratio between reduced signatures. We observe that the overlapping ratio decreases from almost 100% to 1% when the original signatures are reduced to 10-dimension. Later sections also discuss the improvement in moving object linking efficiency due to spatial shrinking.

4 Moving Object Linking

Signature reduction helps to eliminate the curse of dimensionality and subsequently speeds up the cosine similarity calculation. However, another issue to be considered is the high cardinality of the candidate object set $|O| = n$. Recall that for each object, we need to conduct a k-NN search in the candidate set. The naive sequential scan approach is clearly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

infeasible since n will be extremely large in real applications. In this section, we will first explore the possibility of employing existing search and indexing algorithms to solve the moving object linking problem, and then introduce in detail our proposed indexing structure to further improve the linking efficiency.

4.1 Baselines

Baseline 1. Our problem can be regarded as a cosine similarity search problem. As a result, existing approaches to speeding up cosine similarity search, such as *AllPairs* [11], *APT* [12], *MMJoin* [13] and *L2AP* [14], can be easily adjusted to solve this problem. Nevertheless, all these algorithms target the curse of dimensionality, i.e., the time cost of calculating cosine similarity based on full-sized vectors. They reduce the number of compared elements at the beginning of the vectors for early candidate pruning. After signature reduction, we suffer more from the high cardinality of the object set rather than the signature size, making these algorithms inappropriate.

Baseline 2. k-NN search has also been extensively studied in Euclidean space, and various spatial indices have been proposed for speeding up k-NN search in a relatively high-dimensional Euclidean space, such as *R-tree* [18] [19] [20] [26], *k-d tree* [15] [17] [16], *hB-tree* [21] [22], etc. Can we transform the k-NN search problem under cosine similarity into that in the Euclidean space? The answer is positive based on the following theorem:

Theorem 1. Given two vectors $X = \langle x_1, x_2, \ldots, x_d \rangle$ and $Y = \langle y_1, y_2, \ldots, y_d \rangle$ under L_2 normalization, their cosine similarity $\cos(X, Y)$ is negatively correlated to their Euclidean distance $euc(X, Y)$.

Proof: Note that both X and Y have been L_2 normalized to unit length, meaning that $||X|| = \sqrt{\sum x_i^2} = 1$ and $||Y|| = \sqrt{\sum y_i^2} = 1$. Therefore,

$$\cos(X, Y) = \frac{X \cdot Y}{||X|| \times ||Y||} = X \cdot Y = \sum x_i y_i$$

$$\begin{aligned} euc(X, Y) &= \sqrt{\sum (x_i - y_i)^2} \\ &= \sqrt{\sum x_i^2 + \sum y_i^2 - 2 \times \sum x_i y_i} \\ &= \sqrt{2 - 2 \times \cos(X, Y)} \end{aligned}$$

It can be observed that the larger $\cos(X, Y)$ is, the smaller $euc(X, Y)$ will be. In other words, $\cos(X, Y)$ is negatively correlated to $euc(X, Y)$. □

To address the object linking problem, we regard each object's signature $f(o)$ as a d-dimensional point and utilize the existing spatial indices to find the k-nearest neighbors in O that minimize the Euclidean distance $euc(f(q), f(o))$ as the linking candidates for the query object q. However, previous work has revealed that some spatial indices (e.g., R-tree and k-d tree) are practically efficient only when the dimensionality d is not too large (e.g., $d \leq 20$). In a sufficiently high-dimensional situation (e.g., $d > 20$), approximate algorithms, such as LSH [9] [10], are widely-adopted alternatives which can obtain near-optimal results for k-NN search with a high probability.

4.2 Weighted R-tree

In the above baselines, we regard a signature purely as a weighted vector while ignoring the information associated with each element. In fact, the signature is composed of a collection of spatial points in the 2D space. In this section, we present in detail our novel indexing structure which we call *Weighted R-tree (WR-tree)*, and show that better efficiency can be achieved by taking both spatial and weight information into consideration.

4.2.1 Pruning Strategies

We first introduce two pruning strategies which can help to rule out unpromising candidates earlier.

1) Pruning by Spatial Overlapping

Our indexing scheme is based on the locality assumption that each moving object usually travels in a limited region. For example, individuals with a nine-to-five working pattern are always subject to routes between residences and workplaces during weekdays. Taxi drivers, although traveling according to customers' requirements, have the flexibility to choose preferred riding regions. In other words, we can bound an object o's signature, in particular spatial points in the signature, with a minimum bounding rectangle (MBR) denoted as $MBR(o)$. We say $p \in MBR(o)$ if point p is spatially covered by $MBR(o)$, and $MBR(o_1) \cap MBR(o_2) \neq \emptyset$ if two MBRs $MBR(o_1)$ and $MBR(o_2)$ spatially overlap with each other. Note that $MBR(o_1) \cap MBR(o_2) \neq \emptyset$ only specifies the spatial overlapping between two MBRs, which does not necessarily mean the corresponding signatures intersect with each other.

Theorem 2. Given two moving objects o_1 and o_2 and their minimum bounding rectangles $MBR(o_1)$ and $MBR(o_2)$, if $MBR(o_1) \cap MBR(o_2) = \emptyset$, then $sim(o_1, o_2) = 0$.

Accordingly, we can skip calculating the cosine similarity between query q and the candidate object o if their MBRs do not overlap with each other. Checking for MBR overlap is intuitively faster in comparison to calculating cosine similarity. Our statistical analysis about the spatial shrinking effect of CUT signature reduction (i.e., only 1% overlapping ratio between reduced signatures) suggests that the spatial pruning strategy could be very effective for speeding up moving object linking.

Baseline 3. A straightforward way of utilizing this pruning strategy is to calculate the MBRs of all the candidate objects using the R-tree indexing scheme. Given a query MBR, we conduct a *range query* on the R-tree to find all the candidate MBRs overlapping with the query MBR. We then calculate the cosine similarities between these candidates and the query object one by one, and determine k-nearest neighbors based on the ranking of cosine similarities. This approach can speed up k-NN search as it eliminates many unnecessary similarity calculations. Nevertheless, each point in the object signature is associated with not only the spatial information but also the weight. Is it possible to combine both types of information to improve search efficiency?

2) Pruning by Signature Similarity

We introduce a method to further prune the k-NN search by considering the signature similarity. Assume that multiple

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

signatures, $f(o_1), f(o_2), \ldots, f(o_m)$, are aggregated into one signature denoted as $f(o_1, o_2, \ldots, o_m)$ such that:

- Point set of the aggregated signature is a union of point sets of the constituting signatures, i.e., $p \in f(o_1, o_2, \ldots, o_m)$ iff $\exists o \in \{o_1, o_2, \ldots, o_m\}, p \in f(o)$;
- Each point weight in the aggregated signature is the maximum value of the corresponding point weights from the constituting signatures, i.e., $w(p) = \max\{w_{o_1}(p), w_{o_2}(p), \ldots, w_{o_m}(p)\}$.

Theorem 3. Given an aggregated signature $f(o_1, \ldots, o_m)$ and a query signature $f(q)$, if $f(o_1, \ldots, o_m) \cdot f(q) < \theta$, then $\forall o \in \{o_1, o_2, \ldots, o_m\}, sim(o, q) < \theta$.

Proof: Consider any object $o \in \{o_1, o_2, \ldots, o_m\}$. Based on the signature similarity defined in Eq. 3,

$$\begin{aligned} sim(o, q) &= \sum w_o(p) \times w_q(p) \\ &\leq \sum \max\{w_{o_1}(p), w_{o_2}(p), \ldots, w_{o_m}(p)\} \times w_q(p) \\ &= f(o_1, o_2, \ldots, o_m) \cdot f(q) < \theta \end{aligned}$$

This ends the proof of Theorem 3. □

Therefore, we can ignore checking objects $o_1, o_2, \ldots, o_m$ if their aggregated signature is not a promising k-NN candidate for the query object q.

4.2.2 WR-tree structure and k-NN search

The basic idea of our indexing scheme is to cluster the set of candidate objects into several disjoint subsets and aggregate them into multiple hierarchies, so that we can prune unpromising subsets earlier based on both spatial overlapping and signature similarity. To this end, we propose the Weighted R-tree index, which incorporates point weights into the R-tree structure.

Fig. 1 illustrates an example of the WR-tree. Each MBR at the leaf describes a single moving object, while MBRs at higher levels represent the aggregation of all the child nodes. Leaf nodes store three fields: 1) object identifier, 2) object signature, and 3) object MBR. MBRs are represented using two points, e.g., $MBR(o_3) = (a, b)$. Intermediate nodes also store three fields: 1) pointers to child nodes, 2) aggregated signature of child nodes, and 3) MBR corresponding to the aggregated signature. For example, since p_8 occurs in both $f(o_1)$ and $f(o_2)$ with different weights 0.3 and 0.5 respectively, the weight of p_8 in $f(A_3)$ takes the largest value 0.5.

Algorithm 1 describes the overall process of k-NN search on the WR-tree considering both pruning methods. Given a query object q along with its signature and MBR, we search for k-nearest neighbors in the WR-tree using a "best-first" strategy. In particular, we use a priority queue Q to arrange candidate tree nodes in descending order of their signature similarities with q. Note that the signature of a non-leaf node is the aggregated signature whose similarity with $f(q)$ could be larger than 1. The priority queue is initialized with the root node and similarity d (line 2. d is the dimensionality of a signature vector, and hence the upper bound of signature similarity). In each iteration, we pop the top tree node u from Q which has the currently largest $sim(u, q)$ (line 3) and check whether u is a leaf node or an intermediate node. If u is a leaf node, a moving object has been retrieved. We insert

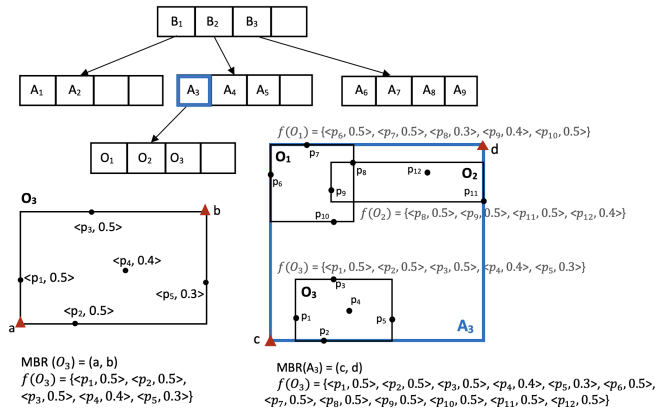


Fig. 1. An example of the WR-tree.

Algorithm 1: k-NN search on WR-tree

Input: T: WR-tree, q: query object
Output: kNN: k-nearest neighbors

1 $kNN = \emptyset, kSim = 0, Q = \emptyset$;
2 $Q.push(\langle root(T), d\rangle)$;
3 **while** $Q \neq \emptyset$ **do**
4 $\quad \langle u, sim(u, q)\rangle = pop(Q)$;
5 $\quad$ **if** $sim(u, q) \leq kSim$ **then**
6 $\quad\quad$ break;
7 $\quad V = child(u)$;
8 $\quad$ **if** $V = \emptyset$ **then**
9 $\quad\quad kNN.insert(\langle u, sim(u, q)\rangle)$;
10 $\quad\quad kSim = kNN[k].sim$;
11 $\quad$ **else**
12 $\quad\quad$ **for** $v \in V$ **do**
13 $\quad\quad\quad$ **if** $MBR(v) \cap MBR(q) \neq \emptyset$ **then**
14 $\quad\quad\quad\quad sim(v, q) = f(v) \cdot f(q)$;
15 $\quad\quad\quad\quad$ **if** $sim(v, q) > kSim$ **then**
16 $\quad\quad\quad\quad\quad Q.push(\langle v, sim(v, q)\rangle)$;
17 **return** kNN;

u into the list of k-nearest neighbors NN if $sim(u, q) > kSim$, where $kSim$ denotes the signature similarity of the k-th neighbor with q, and then update $kSim$ (lines 8-10). If u is an intermediate node, we retrieve all its child nodes and add promising children into the queue for further checking (lines 11-16). During this process, the aforementioned rules are used to identify unpromising child node v and prune the corresponding search branch:

- Pruning by spatial overlapping: If $MBR(v) \cap MBR(q) = \emptyset$, this branch can be pruned without calculating the signature similarity (line 13);
- Pruning by signature similarity: If $sim(v, q) \leq kSim$, all its child nodes are unpromising for k-NN search and can be pruned (line 15).

The k-NN search continues until the priority queue Q is empty (line 3) or the top tree node in Q is unpromising (lines 4-6, all the remaining nodes in Q and their children are unlikely to achieve a similarity larger than $kSim$).

4.2.3 WR-tree Construction and Update

An important factor that needs to be carefully considered when constructing WR-tree is how to hierarchically cluster candidate objects to optimize pruning power. In traditional R-trees, minimization of both *coverage* (i.e., MBRs do not

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

cover too much empty space) and *overlap* (i.e., MBRs do not share too much common space so that fewer subtrees need to be processed during search) is vital to the performance.

In the WR-tree, however, we should also consider the size of the aggregated signature due to the following reasons: 1) The aggregated signature is stored in every internal node, which means we can reduce the storage cost of WR-tree if the size of each aggregated signature is minimized; 2) An upper bound needs to be calculated when pruning an internal node and the corresponding subtree by signature similarity. The upper bound is defined as the cosine similarity between the query object's signature and the aggregated signature stored in the internal node. Therefore, the size of the aggregated signature will affect the computational cost of k-NN search.

When constructing the WR-tree, we consider the following heuristic criteria for optimization:

- Minimize the size of the aggregated signature.
- Minimize the area covered by the MBR.
- Minimize the overlapping between MBRs.

In other words, we further examine the *signature enlargement* when merging two nodes, in addition to area enlargement and overlapping enlargement. A natural way of reducing signature enlargement is to maximize the number of common points between two signatures. As illustrated in Fig. 2, the signature $f(q)$ spatially overlaps with all the three signatures $f(o_1)$, $f(o_2)$ and $f(o_3)$. Although combining $f(q)$ and $f(o_1)$ results in a smaller area enlargement than merging $f(q)$ and $f(o_2)$, the signature enlargement increases on the contrary since $f(q)$ shares less common points with $f(o_1)$ than $f(o_2)$.

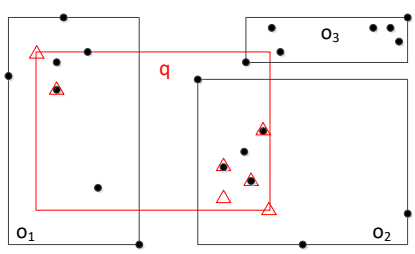


Fig. 2. An example of signature enlargement. Rectangles represent MBRs, dots and triangles represent points contained in the signature. A dot and a triangle correspond to the same point when overlapping.

We propose a bulk-loading algorithm for building WR-tree based on the strategy known as Sort-Tile-Recursive (STR) [27]. It builds the WR-tree in a bottom-up manner where tree nodes are recursively merged into upper levels. The total number of leaf nodes is initialized as $l = \lceil \frac{n}{c} \rceil$, where n is the number of objects and c is the node capacity. We spatially sort the objects based on the longitude (resp. latitude) and divide them into several equal-size partitions with the size of $s = \lceil \frac{n}{l} \rceil$ (resp. $s = \lceil \frac{n}{\sqrt{l}} \rceil$). Objects in each partition are merged into tree nodes. If the bulk size s exceeds the node capacity c, they will be recursively bulk-loaded using the same method. Algorithm 2 depicts our implementation of the *MergeNode()* function. Given a bulk of spatially ordered objects, each tree node is initialized by the first unassigned object (line 10) and the remaining $c - 1$ objects are selected from the bulk by minimizing signature enlargement (i.e., maximizing the number of common points, lines 11-13). Compared with the insertion-based index construction adopted in [28], this bulk-loading based WR-tree performs significantly better, as demonstrated in our experimental results in Section 6.3.2.

Algorithm 2: MergeNode()

```
   Input: L: the list of spatially ordered objects in a bulk; c: node capacity
   Output: N: newly created nodes
1  N = ∅;
2  if |L| < c then
3      node = createNode(L);
4      N.add(node);
5  else
6      n = ⌈|L|/c⌉; // # nodes to be created
7      idx = 0;
8      while |N| < n do
9          node = createNode();
10         node.addElement(L.get(idx));
11         while node.size() < c do
12             e = MaxCommonPoints(node, L);
13             node.addElement(e);
14         N.add(node);
15         Update(idx, L); // get the index of next available element in L
16 return N;
```

Algorithm 3: UpdateSubtree()

```
   Input: node: root of the subtree to be updated; o: object to be inserted
1  Update(node, o);
2  C = child(node);
3  N = MaxCommonPoints(o, C);
4  if |N| > 1 then
5      N = MinArea(o, N);
6      if |N| > 1 then
7          N = MinOverlap(o, N);
8  Randomly select a node from N and regard it as target;
9  // Tree nodes are updated recursively until the leaf
10 if target.level = 1 then
11     Update(target, o);
12 else
13     UpdateSubtree(target, o);
```

A WR-tree is useful in practice only if it can append new objects after construction while maintaining an acceptable arrangement of the tree nodes to maximize pruning power. We adopt an incremental strategy to update the WR-tree. Specifically, when inserting a new object, we traverse down to the leaf node by recursively choosing the most attractive subtree based on the three heuristic criteria discussed above. Algorithm 3 describes our implementation of the *UpdateSubtree()* function, where we select the subtree by maximizing the number of common points between signatures and then minimizing the area and overlapping between MBRs. Tree nodes along the traversal path are updated with respect to the corresponding aggregated signature and MBR. However, we observe that the WR-tree gradually becomes less efficient with more and more new objects being inserted beyond the initial bulk-loading. According to our empirical results in Section 6.3.2, querying a WR-tree that has been updated using incremental insertions for several times is less efficient compared to a newly bulk-loaded tree indexing the same objects. Therefore, a more practical solution would be to periodically rebuild the WR-tree using bulk-loading so that objects are arranged globally and evenly in the tree.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

5 Robustness and Linkability

In this section, we introduce two optimization methods to further improve the accuracy and robustness of object linking. We also discuss the possibility of using our designed signature for trajectory privacy protection.

5.1 Improving Linking Accuracy

The major motivation of signature reduction is to mitigate the curse of dimensionality and thus improve the linking efficiency. However, a drawback of using the reduced signatures is an observed drop in the linking accuracy of the k-NN search results. This is because the reduced signatures essentially encode less information than larger signatures, making it more difficult to link objects. We introduce two optimization schemes to mitigate this drawback without significantly impacting the efficiency. Note that these optimization schemes are generalized and can be applied to any of the aforementioned linking approaches.

1) Re-Ranking Strategy

Naturally, the matched objects are much more likely to be retrieved by a top-k ($k > 1$) NN search than a stricter top-1 NN search. In other words, with the limited information encoded in a reduced signature, the correct linking might appear at a lower rank than the top-1 result, meaning that a further refinement of the k-NN search results is needed. To this end, we introduce a heuristic *re-ranking* strategy to improve the accuracy of stricter linking queries. Specifically, given a query object q, we first retrieve the top-k ($k > 1$) candidate objects using signatures of small size (e.g., d=10), and then re-order those objects by calculating their signature similarities using larger and more informative signatures (e.g., d=500). It is obvious that the pre-filtering phase is quite efficient by k-NN search with small-size signatures, and the extra computational cost is also acceptable by limiting the re-ranking only in the top-k candidates.

2) Stable Marriage Algorithm

Our second optimization scheme extends the classic *stable marriage* problem (SMP) [29]. Given two equal-size datasets, stable marriage aims to find a one-to-one mapping between the elements of each dataset. Elements are mapped based on a given order of preferences for elements in their opposing dataset, and the final mapping of elements must be stable, i.e., no two unpaired elements would rather be paired with each other.

In practice, a guaranteed stable marriage requires a complete pairwise computation of the preference lists for every element, incurring an unacceptable time cost. Additionally, the problem requires two equally sized datasets, which may not necessarily be the case. Hence, traditional algorithms for stable marriage cannot be directly applied to our object linking problem, even though signature similarity can naturally be used to rank preferences. In this work, we modify the stable marriage algorithm to explore the most similar candidate for each object and guarantee a relatively stable matching between two sets of moving objects. As shown in Algorithm 4, our stable marriage method is based on each object's top-k NNs, which have been quickly generated by using any of the aforementioned linking algorithms on the reduced signatures. For each object, its set of top-k NNs along with their signature similarities form a preference list which will be used in the following *proposal* phase (line 8). The process of *proposal* and *engage* repeats until no stable match can be obtained further (lines 3-20). Finally, all the unmatched objects will use the original k-NN results as their linking (lines 21-23). Our empirical results in Section 6.2.4 verify that stable marriage can effectively improve accuracy of the top-ranked object linking, while incurring limited extra computational cost.

Algorithm 4: k-NN Based Stable Marriage

Input: O, O': two sets of objects to be linked together; kNN, kNN': two sets of top-k NNs for each object from both sets;
Output: P: the stable matching pairs between two sets

```
1  P = ∅;
2  Q = ∅; // the set of unmatched objects
3  while O ≠ ∅ do
4      o_i = O.poll();
5      o'_m = kNN_i.poll();
6      if P.contains(o'_m) then
7          o_j = P.get(o'_m);
8          if getRank(o_i, kNN'_m) > getRank(o_j, kNN'_m) then
9              P.add(o_i, o'_m);
10             if kNN_j ≠ ∅ then
11                 O.add(o_j);
12             else
13                 Q.add(o_j);
14         else
15             if kNN_i ≠ ∅ then
16                 O.add(o_i);
17             else
18                 Q.add(o_i);
19     else
20         P.add(o_i, o'_m);
21 // Handle the remaining objects which fail to be matched before
22 for o_i ∈ Q do
23     P.add(o_i, getTopOne(kNN_i));
24 return P;
```

5.2 Trajectory Linkability

Trajectories can disclose highly-sensitive information of an individual, such as personal gazetteers and social relationships. Even when trajectories are anonymized (namely, IDs removed), the possibility of object re-identification through spatiotemporal entity linking, as studied in this work, still exposes a high risk of privacy leaks. Although much effort has been devoted to adapting the existing privacy protection models to trajectory dataset, such as k-anonymity [30] [31], l-diversity and t-closeness [32], differential privacy [33], and plausible deniability [34], they either incur massive computational cost [31] or cannot achieve a satisfactory protection against the linking attack.

In this work, we first explore the extent of linkability, especially how much data is needed to achieve a reliable linking. Our empirical result in Figure 3 shows that signatures derived from about one-week trajectories are enough to identify the majority of individuals ($> 75\%$). Moreover, we study the possibility of addressing this linkability attack using our proposed signature design. We introduce a solution called *signature closure*, which iteratively suppresses the reduced signature (i.e., the top-m TF-IDF weighted points) of a moving object from its historical trace and releases the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

modified trajectories. Our experimental results in Section 6.2.5 demonstrate that objects with modified traces cannot be easily re-identified. It is worth noting that such modification will not affect much of the utility as evidenced by Table 9, since only a very limited amount of information (i.e., a small set of points) is removed from the original trajectories.

6 Experiments

We conduct extensive experiments on a real-world dataset to evaluate the performance of our algorithms. We report the experimental results and analysis in this section.

6.1 Experiment Setting

Dataset. We use a real-life dataset of 359,666,430 GPS points generated by 13,132 taxis in Beijing city over one month. In order for the signature to capture enough information about a taxi's moving pattern, we remove taxis with less than 7,000 points and finally reserve 12,000 taxis. Each taxi is associated with a trace which is a concatenation of all its trajectories in chronological order. We use a commercial map of Beijing to extract road intersections and align traces by trajectory calibration [35]. 181,265 intersections are discovered, which is regarded as the dimensionality of the original signature. We then randomly sample 3000, 6000, 9000 taxis respectively, from the original dataset, to evaluate the performance of moving object linking on different sizes of datasets. We also evaluate our proposals on another publicly available dataset, Geolife [36], which contains check-ins generated by 178 users of the Geolife social networking service in a period of over four years (from April 2007 to October 2011). After pre-processing, 175 users are remained with 20,828,028 GPS points in total. We report experimental results on the Geolife dataset in our technical report [37] due to space limitation.

Evaluation metrics. We divide the original dataset into two parts Q and D. Each taxi exists in both datasets and its trace consists of only half of the original trajectory set. We assign trajectories to Q and D alternately to eliminate the influence of temporal dynamics (e.g., Q and D include all trajectories in odd days and even days respectively, and each contains 15 days of trajectories). For each object q in dataset Q, we conduct a k-NN search in dataset D and check if the same object appears in the top-k neighbors. We run this operation $|Q|$ times, and report the average accuracy.

$$Acc@k = \frac{|Q_k^*|}{|Q|} \tag{4}$$

In Eq. 4, Q_k^* represents the set of query objects that can successfully find themselves in the corresponding top-k results, namely, $Q_k^* = \{q | q \in kNN(q, D)\}$.

Algorithms. Our performance evaluation has two parts: accuracy and efficiency. For accuracy evaluation, we compare the four signature representations introduced in Section 3.1: sequential, temporal, spatial, and spatiotemporal. We then report $Acc@k$ achieved by the original spatial signature and compare it with the signature reduction strategies discussed in Section 3.2: CUT, PCA, and LSH (We control the reduced dimensionality m of LSH by the number of hash functions). We also evaluate the effectiveness of the optimization methods introduced in Section 5.1: re-ranking, stable marriage, and their combined effect. For the efficiency evaluation, we compare WR-tree with all baselines discussed in Section 4, and report the performance of index build and update.

All the above algorithms are implemented in Java, and all the experiments are conducted on a server with two Intel(R) Xeon(R) CPU E5-2630, 10 cores/20 threads at 2.2GHz each, 378GB memory, and Ubuntu 16.04 operating system.

6.2 Effectiveness

6.2.1 Signature Representations

Table 2 shows the accuracy of moving object linking using sequential, temporal, spatial and spatiotemporal signatures respectively on the dataset Q and D with 3000 randomly sampled taxis. We observe that the accuracy of sequential signatures decreases with subsequence length q. Interestingly, the sequential signature is the most effective when $q = 1$ which actually corresponds to a geo-spatial point. This reflects that the sequential feature is not helpful for object linking, despite producing strong empirical results for other applications such as location prediction. Temporal signature performs much worse than the other two counterparts, since it simply extracts a coarse distribution of an object's temporal moving behavior. Although the accuracy increases gradually when a finer granularity (i.e., smaller Δt) is used, the improvement is still insignificant. TF-IDF-based spatial signature is quite effective in modeling an object's traveling behavior, which achieves a 85.5% accuracy when linking objects to their top-1 nearest neighbors, and the accuracy keeps enhancing when k rises. This verifies the importance of spatial features in moving object modeling and linking. As for the spatiotemporal signature, it defeats the temporal signature thanks to the extra spatial information. However, adding temporal information does not improve performance versus a pure spatial signature under any spatial resolution. This is reasonable since important locations in an object's mobility patterns might be visited at different times, making the patterns extremely sparse. Hence, we only consider the spatial signature in the remaining experiments.

6.2.2 Signature Reduction

Table 3 illustrates the accuracy of various signature reduction strategies (i.e., PCA, LSH, and CUT). The original signature contains around 160,000 points, and we reduce it to $m = 10$, 50, 100, 500, 1000 points respectively. We have the following observations from Table 3: 1) Linking accuracy degrades with m due to information loss when we reduce the dimensionality of the original signature to m. 2) CUT achieves consistently larger accuracy than LSH which in turn outperforms PCA. Such superiority is extremely evident when the dimensionality is small ($m \leq 100$). PCA is a traditional dimension reduction algorithm and has been empirically proven to be inferior to LSH in many applications. Although LSH is widely-adopted for approximate nearest neighbor search in high-dimensional space (e.g., multimedia search, gene expression identification, near-duplicate document detection, etc.), a simpler method that cuts the long tail of the signature vector has been demonstrated to be more effective in object linking. 3) Our CUT algorithm can successfully reduce the signature dimensionality from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

TABLE 2
Effectiveness of signature representation ($|D| = 3000$).

Methods	Sequential (q)					Temporal (Δt)							Spatial	Spatiotemporal (# of grids)		
Parameters	1	2	3	4	5	1h	2h	3h	4h	6h	8h	12h	N/A	100^2	200^2	300^2
$Acc@1$	0.855	0.804	0.773	0.737	0.704	0.127	0.123	0.104	0.087	0.042	0.018	0.004	0.855	0.535	0.567	0.583
$Acc@2$	0.904	0.862	0.838	0.811	0.791	0.169	0.167	0.145	0.124	0.074	0.033	0.007	0.904	0.587	0.613	0.630
$Acc@3$	0.928	0.892	0.871	0.848	0.829	0.195	0.186	0.172	0.150	0.092	0.046	0.009	0.928	0.612	0.640	0.651
$Acc@4$	0.940	0.913	0.891	0.872	0.854	0.216	0.205	0.198	0.174	0.113	0.057	0.011	0.940	0.632	0.659	0.681
$Acc@5$	0.948	0.924	0.905	0.887	0.869	0.233	0.220	0.216	0.192	0.131	0.071	0.013	0.948	0.647	0.673	0.693

TABLE 3
Effectiveness of signature reduction ($|D| = 3000$).

Methods	PCA					LSH					CUT					Original
m	10	50	100	500	1000	10	50	100	500	1000	10	50	100	500	1000	160,000
$Acc@1$	0.007	0.050	0.113	0.542	0.697	0.046	0.476	0.638	0.795	0.824	0.806	0.827	0.831	0.836	0.838	0.855
$Acc@2$	0.012	0.088	0.187	0.686	0.801	0.079	0.542	0.705	0.847	0.870	0.866	0.877	0.880	0.885	0.886	0.904
$Acc@3$	0.018	0.123	0.243	0.765	0.846	0.097	0.577	0.731	0.872	0.893	0.893	0.903	0.907	0.913	0.916	0.928
$Acc@4$	0.023	0.150	0.289	0.809	0.875	0.118	0.597	0.748	0.891	0.912	0.906	0.919	0.920	0.928	0.929	0.940
$Acc@5$	0.031	0.176	0.333	0.835	0.892	0.130	0.617	0.760	0.900	0.924	0.917	0.929	0.930	0.937	0.939	0.948

160,000 to 10 at a slight cost of linking accuracy ($< 5\%$). This verifies the possibility of identifying a moving object based on several geo-spatial locations in its traveling history, which is consistent with the phenomenon observed in [1]. We will use reduced signatures obtained by CUT algorithm with $m = 10$ by default for the following experiments.

6.2.3 Quantitative Signature Metrics

To further understand the proposed spatial signature, we also evaluate the respective contributions of the quantitative signature metrics, i.e., commonality and unicity. The results are reported in Table 4. We can observe that the TF-based method performs consistently better than the IDF-based counterpart (avg. 3% and 10% accuracy improvement for reduced and original signature respectively), which reflects that commonality of a signature is more important than unicity for modeling an individual's movement behavior. Meanwhile, these two criteria perfectly complement each other, resulting in a much higher accuracy of object linking when combining them via the TF-IDF measure. Such a phenomenon is especially notable for the reduced signature (avg. 70% accuracy improvement) since only the most important locations in the historical trace are preserved in the reduced signature. This verifies that a signature should be both representative (i.e., high commonality) and distinctive (i.e., high unicity) to effectively identify a moving object.

TABLE 4
Contributions of commonality and unicity ($|D| = 3000$).

Signature	Reduced ($m = 10$)			Original		
Weighting Strategy	TF	IDF	TF-IDF	TF	IDF	TF-IDF
$Acc@1$	0.119	0.085	0.806	0.721	0.592	0.855
$Acc@2$	0.161	0.126	0.866	0.765	0.666	0.904
$Acc@3$	0.191	0.154	0.893	0.790	0.705	0.928
$Acc@4$	0.214	0.173	0.906	0.810	0.735	0.940
$Acc@5$	0.231	0.187	0.917	0.823	0.750	0.948

6.2.4 Accuracy and Robustness

Here we evaluate the robustness of our linking algorithm and the performance of the two optimization schemes (i.e., re-ranking (RR) and stable marriage (SM)). Table 5 reports the linking accuracy for different dataset sizes (i.e., number of objects). The accuracy is very high for all datasets, which clearly demonstrates the stability of our signature design.

TABLE 5
Effectiveness on different dataset size.

Signature	Reduced ($m = 10$)				Original			
$\|D\|$	3000	6000	9000	12000	3000	6000	9000	12000
$Acc@1$	0.806	0.767	0.755	0.754	0.855	0.831	0.825	0.829
$Acc@2$	0.866	0.837	0.825	0.825	0.904	0.879	0.874	0.874
$Acc@3$	0.893	0.867	0.860	0.858	0.928	0.900	0.894	0.896
$Acc@4$	0.906	0.884	0.877	0.877	0.940	0.914	0.908	0.910
$Acc@5$	0.917	0.893	0.890	0.888	0.948	0.922	0.916	0.917

TABLE 6
Effectiveness of two optimization schemes on different dataset sizes.

Scheme	×	RR				SM
m	10	50	100	500	original	10
$\|D\| = 3000$	0.806	0.823	0.828	0.834	0.841	0.845
$\|D\| = 6000$	0.767	0.792	0.797	0.805	0.814	0.813
$\|D\| = 9000$	0.755	0.789	0.794	0.799	0.811	0.802
$\|D\| = 12000$	0.754	0.791	0.798	0.803	0.815	0.803

TABLE 7
Extra time cost (s) of optimization schemes on different dataset sizes.

Scheme	×	RR				SM
m	10	50	100	500	original	10
$\|D\| = 3000$	0.164	0.026	0.049	0.255	13.886	0.190
$\|D\| = 6000$	0.464	0.057	0.108	0.499	28.684	0.497
$\|D\| = 9000$	0.898	0.118	0.204	1.131	40.732	0.972
$\|D\| = 12000$	1.738	0.141	0.246	1.576	52.258	1.804

Table 6 reports the accuracy improvement using RR and SM for $Acc@1$. For RR, we first use 10-dim signatures to quickly find the top-k candidates for each object, then the k candidates are re-ranked using larger signatures of sizes $50, 100, 500$ and the full-size for a more accurate similarity comparison. Algorithm 4 is used to apply stable marriage reshuffle ($k = 5$ for all experiments here). From Table 6

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

we observe that two strategies are very effective. Compared with the result of only using 10-dim signatures, the SM process leads to a 4.5% increase of linking accuracy on average, while the RR improvement depends on different sizes of signatures that are used (ranging from 1.7% to 6.1%). The additional time costs for applying RR and SM are shown in Table 7. To put the cost for re-ranking into context, full-sized signatures can have an average of $160,000$ points and the re-ranking stage with full-sized signatures takes much longer time with little accuracy gain (as shown above). The revised SM process is slightly superior over the RR method, since it achieves a larger accuracy improvement with a comparable time cost ($m = 500$ for RR). Overall, it is worthwhile to spend less than two more seconds for a nearly 5% accuracy increase (in the case of $|D| = 12000$).

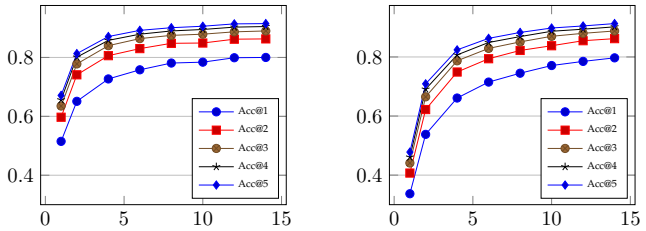
Furthermore, we examine whether the combination of RR and SM can achieve even better accuracy. Table 8 shows that applying SM on the sets of RR-improved top-5 candidates maintains a significant improvement of about 4% for $Acc@1$. As k increases, the accuracy gain diminishes. Therefore, combining RR and SM is a good strategy to improve linking accuracy, especially when the top-1 candidate is of the main interest.

TABLE 8
Accuracy using RR and SM with different m ($|D| = 3000$).

Scheme	SM	RR			Combination		
m	10	50	100	500	50	100	500
$Acc@1$	0.845	0.823	0.828	0.834	0.864	0.869	0.874
$Acc@2$	0.883	0.873	0.875	0.876	0.898	0.901	0.902
$Acc@3$	0.904	0.898	0.896	0.901	0.908	0.907	0.917
$Acc@4$	0.915	0.909	0.910	0.911	0.916	0.919	0.921
$Acc@5$	0.923	0.917	0.917	0.917	0.923	0.925	0.926

6.2.5 Linkability and Signature Sensitivity

Now, we examine the impact of information size on linkability: how much data is needed for reliable linking? The trajectories in query set Q are limited to d days of data, from 1 to 15 days (full data); Linking is performed with a constant full 15-day dataset (Figure 3-a) and a similarly limited data set using d days of trajectories (Figure 3-b). The d days are chosen at random and the average accuracy is reported for multiple runs. As depicted, the linking accuracy increases significantly with the amount of data, and the accuracy is significantly higher when searching on the full dataset. The accuracy becomes more robust when $d \geq 7$ in both cases. In other words, the signatures become sufficient to guarantee high linking accuracy with only one week's data.



(a) d query days vs 15 days (b) d days vs d days

Fig. 3. Effectiveness on different information size ($|D| = 3000, m = 10$).

While the signatures proposed in this paper can achieve highly accurate linking, it is necessary to understand the importance of the points in the signature closure (Section 5.2) by examining if objects can still be linked after those points in the signatures are removed. Table 9 reports the linking accuracy using the 10-dim signatures. In the 1st round, the signatures are generated using the entire dataset; in the 2nd round, a new signature for each object is generated using the dataset with the points in the previous signatures removed; and so on. In each iteration, we test both the basic "$Acc@1$" and the "Best $Acc@5$" which is derived from the combination of re-ranking and stable marriage with 500-dim signatures (i.e., the best linking accuracy we can achieve). In addition, we also measure the data *utility* affected by removing the top-10 points from the signatures by calculating the average percentage of data remaining, as well as the average overlapping of objects' MBRs and the average percentage of cells in a uniform grid covered by the trajectories before and after signature points are removed (we use 'small' and 'large' grids which are of 85×85 and 423×423 m^2, respectively).

From Table 9 we observe a dramatic decrease of linking accuracy each time when the points in the signatures from the previous round are removed, while the data utility indicators remain to be very high. After removing the signatures for just one round (i.e., removing 10 points for each object which has an average of 30,000 points originally), the best linking accuracy drops from an initial 92.6% to an unusable 45%. Results in Table 9 clearly illustrate that the signature points of an object generated using our approach are highly representative and discriminative to the object. These points are suitable as signature representations for object linking. Conversely, it is a very promising approach to protect trajectory data by removing a small number of points (i.e., the points in the signature closure, or the first a few rounds of it) such that the reidentification attacks can be prevented to a large extent. Trajectory privacy protection is beyond the scope of this paper, but we present some interesting and exciting insights here for new ways to protect privacy of trajectory data. By the suppression or anonymization of a small number of points in the signature closure, we can achieve strong results for preventing link-based attacks at a very limited cost of data utility.

TABLE 9
Signature sensitivity ($|D| = 3000, m = 10$).

Round	Linking Accuracy		Data Remain	MBR Overlap	Grid Coverage	
	$Acc@1$	Best $Acc@5$			Large	Small
1	0.806	0.926	0.978	0.999	0.994	0.961
2	0.140	0.450	0.969	0.999	0.989	0.944
3	0.050	0.187	0.960	0.999	0.984	0.928
4	0.028	0.114	0.953	0.998	0.980	0.913
5	0.020	0.078	0.947	0.998	0.975	0.898

6.3 Efficiency

6.3.1 Object Linking Algorithms

Table 10 reports the efficiency of object linking algorithms on datasets of various sizes. L2AP takes the longest time for object linking (outperformed even by the naive linear

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

scan), though it is considered as an efficient approach to cosine similarity search. It is designed for a really high-dimensional space (e.g., million-level), hence is unsuitable for handling reduced signatures as it wastes a large amount of time for excessive bound checking and index construction. LSH is also proposed for speeding up k-NN search in a high dimension (e.g., multimedia data). It only achieves a minor performance improvement compared to the linear scan when signature dimension is reduced to $m = 10$. Overall, our approach works very well by transforming the linking problem to 2D space and utilize R-tree and spatial overlapping for search pruning, as illustrated in 2D R-tree and WR-tree. More importantly, WR-tree outperforms 2D R-tree by a large margin. This verifies the effectiveness of both pruning strategies introduced in this work, namely spatial overlapping and signature similarity. Table 10 also shows the impact of dataset size $|D|$. As expected, the linking time rises with $|D|$, but when the WR-tree is used the increase is quite small ($\approx 0.6s$ when $|D|$ ranges from 3000 to 12000).

TABLE 10
Total time cost (s) of different linking algorithms ($m = 10$, $k = 1$).

$\|D\|$	Linear	L2AP	LSH	R-tree	WR-tree
3000	2.269	3.090	1.769	0.364	0.059
6000	8.182	14.557	6.652	1.518	0.220
9000	19.733	36.541	15.642	3.597	0.387
12000	27.183	70.440	38.131	7.206	0.678

6.3.2 WR-Tree Construction and Update

Recall that we design a new criterion to optimize WR-tree (i.e., signature enlargement), which motivates the introduction of bulk-loading and incremental algorithms for WR-tree construction and update, respectively. Table 11 examines whether the efficiency of WR-tree can be further improved by considering signature enlargement in index construction. We can see that both the linking cost and index size are slightly reduced when the aggregated signature sizes at internal tree nodes are minimized by maximizing the number of common points between signatures. Such improvement increases with the growth of the signature size m. This is because larger signatures will increase not only the cost of similarity calculation but also the possibility of spatial overlapping (recall that our CUT dimension reduction has a natural side effect of "spatial shrinking"). Table 11 also empirically demonstrates the superiority of the bulk-loading index construction approach compared to the insertion-based algorithm adopted in [28], with the linking efficiency improved at least two-fold thanks to a global optimization of the WR-tree structure. Although the index construction time also increases if signature enlargement is incorporated into the bulk-loading algorithm, it is still acceptable since the index is usually constructed only once.

We then examine the efficiency of WR-tree updates. The base WR-tree is built for the dataset with 9,000 objects. The index update costs and the linking efficiency are studied by repeatedly inserting 500 new objects one by one. Table 12 presents the cost to rebuild the WR-tree using the bulk-loading method, the average cost of each update (i.e., appending one object), and the total time of object linking. As expected, the update cost is extremely small compared with the cost of rebuilding the WR-tree from scratch. It verifies the effectiveness of our proposed incremental tree maintenance. A slight decrease in both the tree update time and the linking cost can be observed when the new criterion of signature enlargement is considered in index update. Compared with a newly-built tree, the linking cost of the updated tree increases due to the imperfect tree structure caused by the incremental updates. However, WR-tree is still far superior over other linking methods as shown in Table 10 in terms of the overall linking performance. Clearly, incremental WR-tree maintenance is effective while a periodic index reconstruction is still necessary. This fact provides a solid base for balancing system throughput and query response time when applying the WR-tree in practice.

TABLE 11
Performance of WR-tree construction w.r.t build time (s), linking time (s) and index size (M) ($|D| = 3000, k = 1$).

Build Criteria	Spatial Factors			# Common Points		
Cost	Build	Link	Size	Build	Link	Size
$m = 10$	0.078	0.077	3.650	1.278	0.059	3.571
$m = 50$	0.332	0.681	15.434	7.501	0.447	14.892
$m = 100$	0.604	1.759	29.067	15.694	1.232	27.642

TABLE 12
Efficiency (s) of the newly built WR-tree and the updated WR-tree with different update criteria ($m = 10, k = 1$).

	Newly Built		Spatial Factors		# Common Points	
$\|D\|$	Build	Link	Per Update	Link	Per Update	Link
9500	9.642	0.399	0.0099	0.425	0.0096	0.404
10000	9.879	0.436	0.0107	0.477	0.0104	0.451
10500	10.062	0.471	0.0115	0.512	0.0113	0.498
11000	10.107	0.528	0.0135	0.623	0.0119	0.592
11500	10.278	0.566	0.0157	0.695	0.0126	0.664
12000	10.455	0.609	0.0175	0.824	0.0131	0.794

7 Related Work

In this section, we briefly summarize the existing work in several research areas that are related to our work.

Trajectory pattern mining: The existing works on trajectory pattern mining can be divided into three categories based on different definitions of patterns. *Sequential* pattern mining [4] [5] identifies a common sequence of locations traveled by a certain number of objects within a similar timeslot. Another research branch [6] [7] [38] tries to discover *periodic* activity patterns from the movement history, which are then used for predicting the future behavior of a moving object. Other works attempt to detect *collective* moving patterns, namely, a group of objects that always travel together for a certain period, such as convoy [39], swarm [2], traveling companion [40], and gathering [3], etc. Unlike the described traditional research, our work focuses on extracting patterns that are both common and unique to a moving object to support object identification.

Cosine similarity search: Given two sets of weighted vectors and a threshold, the cosine similarity search finds all vector pairs whose cosine similarity exceeds the threshold. Existing work mainly focuses on quickly locating all necessary candidate pairs by checking only a few elements at

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

the beginning of the vectors, which is well-known as the *pruning-verification* framework. The AllPairs [11] algorithm avoids computing similarity for unpromising vectors by exploiting a dynamically-constructed inverted index. Extensions to the AllPairs algorithm, such as APT [12], MMJoin [13] and L2AP [14], further improve the efficiency by introducing tighter similarity bounds. Alternatives also exist that find approximate answers for cosine similarity search using locality sensitive hashing (LSH) [9] [10]. However, the semantics of vector elements are ignored in these methods, which limits the efficiency in our problem. We show in this work how the geographical information in object signatures can be exploited to improve object linking efficiency.

Spatial indexing: Spatial index is designed for organizing spatial objects and optimizing a wide range of spatial queries. Various indexing structures have been proposed in the literature, including Quadtree [41], R-tree [18] [19] [20] [42] [26] [43], k-d tree [15] [17] [16], hB-tree [21] [22], etc. Specifically, R-tree [18] is a dynamic and balanced tree structure that organizes minimum bounding rectangles (MBRs) of spatial objects into leaf nodes and groups nearby MBRs into internal nodes. The search algorithm determines whether or not to search inside a subtree by checking MBRs. A majority of nodes are pruned in this way, reducing I/O cost. The construction method of R-tree, in particular how to group MBRs into intermediate nodes, is indispensable to its search performance in practice. Most of the R-tree variants, e.g., R+ tree [19], R*-tree [20], Hilbert R-tree [42], X-tree [26], PR-tree [43], etc., target at improving the merging strategy. In this work, we extend the R-tree structure to combine both spatial and weight information of each point for more efficient moving object linking.

Trajectory privacy: Privacy-preserving trajectory publication aims to release individuals' moving trajectories without leaking their sensitive information. It has been extensively studied in the literature, adapting various classic data privacy protection models to trajectory data. k-anonymity algorithms (e.g., W4M [30] and GLOVE [31]) cluster and merge trajectories such that a trajectory is indistinguishable with at least other k-1 trajectories. l-diversity and t-closeness [32] further extend k-anonymity to prevent semantic attack by considering sensitive attribute value distributions in trajectories. Differential privacy (e.g., DPT [33]) and plausible deniability [34] insert random noises into trajectories, ensuring that the presence of an entity or trajectory in a dataset can only be observed for a controlled amount of certainty. Besides, some ad-hoc privacy models, such as dummy [44] and mix-zone [45], have also been applied to anonymize sensitive information for trajectory data. In this work, we show that the TF-IDF-weighted signatures are highly effective for protecting trajectory privacy, especially the linkability attack, through the signature closure concept.

8 Conclusion

In this paper, we have studied the problem of spatiotemporal entity linking based on trajectories. Different signature representation strategies have been examined for their ability to capture the unique characteristics of trajectory data. The linking problem is formalized as a k-NN query based on signature similarity. A comprehensive suite of techniques have been developed, including signature reduction and WR-tree indexing with update support. Two optimization schemes (i.e., re-ranking and stable marriage) are introduced to enhance the linking accuracy. An empirical study on a large taxi dataset demonstrates significantly better accuracy and efficiency of our approach than the state-of-the-art methods. In the future, we will extend our research to heterogeneous datasets over a long period of time to support cross-domain spatiotemporal entity linking. The location privacy projection issue will also be further investigated.

Acknowledgment

This work was partially supported by the Australian Research Council (Grant No. DP200103650 and LP180100018).

References

[1] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific reports*, vol. 3, p. 1376, 2013.

[2] Z. Li, B. Ding, J. Han, and R. Kays, "Swarm: Mining relaxed temporal moving object clusters," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 723–734, Sep. 2010.

[3] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang, "On discovery of gathering patterns from trajectories," in *ICDE '13*, April 2013, pp. 242–253.

[4] H. Cao, N. Mamoulis, and D. W. Cheung, "Mining frequent spatio-temporal sequential patterns," in *ICDM '05*, Nov 2005.

[5] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *KDD '07*, New York, NY, USA, 2007, pp. 330–339.

[6] H. Cao, N. Mamoulis, and D. W. Cheung, "Discovery of periodic patterns in spatiotemporal sequences," *IEEE TKDE*, vol. 19, no. 4, pp. 453–467, April 2007.

[7] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye, "Mining periodic behaviors for moving objects," in *KDD '10*, New York, NY, USA, 2010, pp. 1099–1108.

[8] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[9] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *STOC '98*, New York, NY, USA, 1998, pp. 604–613.

[10] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB '99*, San Francisco, CA, USA, 1999, pp. 518–529.

[11] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," in *WWW '07*, New York, NY, USA, 2007, pp. 131–140.

[12] A. Awekar and N. F. Samatova, "Fast matching for all pairs similarity search," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1, Sept 2009, pp. 295–300.

[13] D. Lee, J. Park, J. Shim, and S.-g. Lee, "An efficient similarity join algorithm with cosine similarity predicate," in *Proceedings of the 21st International Conference on Database and Expert Systems Applications: Part II*, ser. DEXA '10, Berlin, Heidelberg, 2010, pp. 422–436.

[14] D. C. Anastasiu and G. Karypis, "L2AP: Fast cosine similarity search with prefix l-2 norm bounds," in *ICDE '14*, March 2014, pp. 784–795.

[15] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[16] J. T. Robinson, "The K-D-B-tree: A search structure for large multidimensional dynamic indexes," in *SIGMOD '81*, New York, NY, USA, 1981, pp. 10–18.

[17] J. L. Bentley, "Multidimensional binary search trees in database applications," *IEEE Transactions on Software Engineering*, vol. SE-5, no. 4, pp. 333–340, July 1979.

[18] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *SIGMOD '84*, New York, NY, USA, 1984, pp. 47–57.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3036633, IEEE Transactions on Knowledge and Data Engineering

[19] T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R$^+$-tree: A dynamic index for multi-dimensional objects," in *VLDB '87*, San Francisco, CA, USA, 1987, pp. 507–518.

[20] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," in *SIGMOD '90*, New York, NY, USA, 1990, pp. 322–331.

[21] D. B. Lomet and B. Salzberg, "The hB-tree: A multiattribute indexing method with good guaranteed performance," *ACM Trans. Database Syst.*, vol. 15, no. 4, pp. 625–658, Dec. 1990.

[22] G. Evangelidis, D. Lomet, and B. Salzberg, "The hB$^\pi$-tree: A multi-attribute index supporting concurrency, recovery and node consolidation," *The VLDB Journal*, vol. 6, no. 1, pp. 1–25, Feb. 1997.

[23] Y. Tang, L. H. U, Y. Cai, N. Mamoulis, and R. Cheng, "Earth mover's distance based similarity search at scale," *Proc. VLDB Endow.*, vol. 7, no. 4, pp. 313–324, Dec. 2013.

[24] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *FOCS '06*, Oct 2006, pp. 459–468.

[25] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian, "Super-bit locality-sensitive hashing," in *NIPS '12*, USA, 2012, pp. 108–116.

[26] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The X-tree: An index structure for high-dimensional data," in *VLDB '96*, San Francisco, CA, USA, 1996, pp. 28–39.

[27] S. T. Leutenegger, M. A. Lopez, and J. Edgington, "STR: A simple and efficient algorithm for r-tree packing," in *ICDE '97*. IEEE, 1997, pp. 497–506.

[28] F. Jin, W. Hua, J. Xu, and X. Zhou, "Moving object linking based on historical trace," in *ICDE '19*. IEEE, 2019, pp. 1058–1069.

[29] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[30] O. Abul, F. Bonchi, and M. Nanni, "Anonymization of moving objects databases by clustering and perturbation," *Information systems*, vol. 35, no. 8, pp. 884–910, 2010.

[31] M. Gramaglia and M. Fiore, "Hiding mobile traffic fingerprints with glove," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, 2015, pp. 1–13.

[32] Z. Tu, K. Zhao, F. Xu, Y. Li, L. Su, and D. Jin, "Protecting trajectory from semantic attack considering k-anonymity, l-diversity, and t-closeness," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 264–278, 2018.

[33] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava, "DPT: differentially private trajectory synthesis using hierarchical reference systems," *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1154–1165, 2015.

[34] V. Bindschaedler and R. Shokri, "Synthesizing plausible privacy-preserving location traces," in *2016 IEEE Symposium on Security and Privacy*, 2016, pp. 546–563.

[35] H. Su, K. Zheng, H. Wang, J. Huang, and X. Zhou, "Calibrating trajectory data for similarity-based analysis," in *SIGMOD '13*, New York, NY, USA, 2013, pp. 833–844.

[36] Y. Zheng, H. Fu, X. Xie, W.-Y. Ma, and Q. Li, *Geolife GPS trajectory dataset - User Guide*, July 2011.

[37] F. Jin, W. Hua, T. Zhou, J. Xu, M. Francia, M. E. Orlowska, and X. Zhou, "Trajectory-based spatiotemporal entity linking," *Technical Report*, October 2020.

[38] Z. Li, J. Wang, and J. Han, "Mining event periodicity from incomplete observations," in *KDD '12*, New York, NY, USA, 2012, pp. 444–452.

[39] H. Jeung, H. T. Shen, and X. Zhou, "Convoy queries in spatio-temporal databases," in *ICDE '08*, vol. 00, 04 2008, pp. 1457–1459.

[40] L. A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C. C. Hung, and W. C. Peng, "On discovery of traveling companions from streaming trajectories," in *ICDE '12*, April 2012, pp. 186–197.

[41] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, Mar 1974.

[42] I. Kamel and C. Faloutsos, "Hilbert R-tree: An improved r-tree using fractals," in *VLDB '94*, San Francisco, CA, USA, 1994, pp. 500–509.

[43] L. Arge, M. D. Berg, H. Haverkort, and K. Yi, "The priority R-tree: A practically efficient and worst-case optimal R-tree," *ACM Trans. Algorithms*, vol. 4, no. 1, pp. 9:1–9:30, Mar. 2008.

[44] X. Liu, J. Chen, X. Xia, C. Zong, R. Zhu, and J. Li, "Dummy-based trajectory privacy protection against exposure location attacks," in *International Conference on Web Information Systems and Applications*, 2019, pp. 368–381.

[45] X. Liu, H. Zhao, M. Pan, H. Yue, X. Li, and Y. Fang, "Traffic-aware multiple mix zone placement for protecting location privacy," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 972–980.

Fengmei Jin received her Bachelor of Engineering from Sun Yat-Sen University in 2016 and Master of Engineering from Renmin University of China in 2019. Currently, she is a PhD candidate at The University of Queensland. Her research interests include spatiotemporal databases, pattern mining, and data integration.

Wen Hua is a Lecturer at The University of Queensland. She received her PhD and Bachelor degrees in computer science from Renmin University of China in 2015 and 2010, respectively. Her main research interests include database systems, information extraction, data integration, and spatiotemporal data management.

Thomas Zhou received his Bachelor of Science (Computer Science) degree with First Class Honors from The University of Queensland in 2019. He is a Research Assistant at The University of Queensland.

Jiajie Xu received the MS degree from The University of Queensland in 2006 and the PhD degree from the Swinburne University of Technology in 2011. He is currently an Associate Professor with the School of Computer Science and Technology, Soochow University, China. His research interests include spatiotemporal database systems, big data analytics, mobile computing, and recommendation systems.

Matteo Francia is a PhD candidate in Computer Science at The University of Bologna, Italy. He was a visiting scholar at The University of Queensland in 2019. He received the MSc and BSc with honors from the University of Bologna in 2017 and 2014, respectively. His research focuses on analytics of unconventional data, with particular reference to trajectory, social, and sensory data.

Maria E Orlowska is a Professor at Polish-Japanese Academy of Information Technology in Warsaw, Poland. She was Professor of Information Systems at The University of Queensland from 1988 to 2016. She is a Fellow of the Australian Academy of Science. Her main research interests include databases and business IT systems with a focus on modeling and enforcement issues of business processes.

Xiaofang Zhou is a Professor of Computer Science at The University of Queensland. He received his BSc and MSc in Computer Science degrees from Nanjing University in 1984 and 1987,respectively, and his PhD in Computer Science from UQ in 1994. His research interests include spatial and multimedia databases, high performance query processing, data mining, data quality management, and machine learning. He is a Fellow of IEEE.